# **Розв’язування олімпіадних задач**

[Задача 1. «Повороти» 1](#_Toc469433769)

[Задача 2. «Одиниці» 2](#_Toc469433770)

[Завдання 3. Трикутне число 3](#_Toc469433771)

[Задача 4. «Нафтові плями» 4](#_Toc469433772)

[Задача 5. «Прямокутники» 6](#_Toc469433773)

[Задача 6. «Квадрат» 6](#_Toc469433774)

[Завдання 7. «Ламана» 7](#_Toc469433775)

[Задача 8. «Білі плями» 9](#_Toc469433776)

[Задача 9. Цегляна стіна 11](#_Toc469433777)

[Задача 10. Порядок 12](#_Toc469433778)

[Задача 11. Код Грея 14](#_Toc469433779)

[Задача 12. Паліндроми 16](#_Toc469433780)

[Задача 13. Рядки 19](#_Toc469433781)

[Задача 14. Сума 21](#_Toc469433782)

[Задача 15. Квадратний корінь 24](#_Toc469433783)

[Задача 16. MATCHES 28](#_Toc469433784)

[Задача17. POINT 29](#_Toc469433785)

[Задача 18. POLYGON 31](#_Toc469433786)

[Задача 19. Точка 35](#_Toc469433787)

[Задача 20. Стіл 35](#_Toc469433788)

[Задача 21. Максимум 37](#_Toc469433789)

[Задача 22. Нумеролог 38](#_Toc469433790)

[Задача 23. Спіраль 39](#_Toc469433791)

## Задача 1. «Повороти»

Діти заблукали в лісі. Вийшовши з деякої точки з координатами (x;y) вони зробивши N однакових поворотів через однакову кількість метрів повернулися в ту ж саму точку. Визначити кут на який вони відхилялись при кожному повороті.

|  |  |
| --- | --- |
| TURN.DAT | TURN.SOL |
| 0 01 | 180 |

Приклад файлу

*0 0 – координати початкової точки, 1 – кількість поворотів, 180 – кут в градусах на який вони повернули.*

**Розв’язок**

Сума кутів опуклого N-кутника =180\*(N-2).

Кут многокутника $\frac{180∙(N-2)}{N}$.

Кутом повороту в задачі є суміжний кут до кута многокутника

$$180-\frac{180∙(N-2)}{N}=180\left(1-\frac{N-2}{N}\right)=180\left(\frac{N}{N}-\frac{N-2}{N}\right)=180\left(\frac{N-\left(N-2\right)}{N}\right)=180\left(\frac{2}{N}\right)=\frac{360}{N}.$$

В задачі не враховується перший поворот тому кут рівний $\frac{360}{N+1}.$

**Програма**

|  |
| --- |
| C++ |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("turn.dat");ofstream out("turn.sol");int \_tmain(int argc, \_TCHAR\* argv[]){int x,y,n,k;inp>>x>>y>>n;k=360/(n+1);out<<k<<"\n";} |

## Задача 2. «Одиниці»

*Умова.* Дано ціле число *I* записане в десятковій системі числення*.*

*Завдання.* Написати програму ONE.\*, яка порахує кількість одиниць в його двійковому записі.

*Вхідні дані.* Вхідний текстовий файл ONE.DAT містить в єдиному число *I*.

*Вихідні дані.* Вихідний текстовий файл ONE.SOL містить єдине ціле число – кількість одиниць.

|  |  |
| --- | --- |
| ONE.DAT | ONE.SOL |
| 7 | 3 |

Приклади файлів

* **Розв’язок**
* Перевести десяткове число в двійкове поділом на 2 з остачею. При переведенні перевіряти остачу і рахувати кількість одиниць.
* **Програма**

|  |
| --- |
| * C++
 |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("one.dat");ofstream out("one.sol");void main() {int n,k;inp>>n;k=0;while (n>0){if (n%2==1)k++;n=n/2;}out<<k<<"\n";} |

## Завдання 3. Трикутне число

**Трикутне число** — це число кружечків, які можуть бути розставлені у формі рівностороннього трикутника:

 *Т2=3 Т3=6*

Послідовність трикутних чисел *Tn* для *n* = 0, 1, 2, 3… починається так: 0, 1, 3, 6,…

Напишіть програму, яка знаходить ***N-е*** трикутне число.

***Формат вхідних даних:*** у єдиному рядку вхідного файлу ***triangle.in*** записане одне число ***N (0 ≤ N ≤109).***

***Формат вихідних даних:*** у перший рядок вихідного файлу ***triangle.out*** виведіть ***N-е*** трикутне число.

***Приклад вхідних та вихідних даних:***

|  |  |
| --- | --- |
| ***triangle.in*** | ***triangle.out*** |
| 1 | 1 |
| 5 | 15 |

**Розв’язок**

Обчислити суму чисел 1+2+3+…+N.

Спосіб 1. Суматор в циклі.

Спосіб 2. Формула суми арифметичної прогресії $S=\frac{A\_{n}+A\_{1}}{2}n=\frac{n+1}{2}n$ .

**Програма**

|  |
| --- |
| C++ |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("triangle.in");ofstream out("triangle.out");void main() {int n,s;inp>>n;s=0;for(int i=1;i<=n;i++)s=s+i;out<<s<<"\n";} |

## Задача 4. «Нафтові плями»

*Умова.* Після аварії на морській нафтовій свердловині в океан вилилося багато нафти. Вона розтеклася по воді, після чого утворилася певна кількість нафтових плям. Для ліквідації наслідків аварії було створено штаб з координації дій. Співробітники штабу зберігають інформацію про плями в комп'ютері у вигляді матриці розмірністю *M* x*N*. Комірка матриці містить 0, якщо нафтова пляма в цих координатах відсутня та 1, якщо наявна (2*≤ M, N ≤ 100)*. У матриці комірки плям не можуть дотикатися одна до одної ні сторонами, ні кутами.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |

*Завдання.* Для полегшення ліквідації наслідків аварії потрібно написати програму OIL.\*, яка знаходитиме загальну кількість плям та кількість плям з однаковою площею.

*Вхідні дані.* Вхідний текстовий файл OIL.DAT містить в першому рядку два числа *M* та*N,* далі слідують *M* рядків, у кожному по *N* цілих чисел розділених пропусками – елементи матриці.

*Вихідні дані.* Вихідний текстовий файл OIL.SOL містить у першому рядку ціле число *k -* загальну кількість плям, далі у кожному з рядів міститься по два числа, перше – площа плями, друге – їх кількість. Дані посортувати по площах в порядку зростання.

Приклади файлів

|  |  |
| --- | --- |
| OIL.DAT | OIL.SOL |
| 5 51 0 1 0 00 0 1 1 01 0 0 0 01 0 0 0 11 0 1 0 1 | 51 22 13 2 |

**Розв’язок**

Скористуватися рекурсивним методом зафарбовування в інший колір (наприклад 2) всі сусідні точки до i,j які рівні 1

|  |  |  |
| --- | --- | --- |
|  | **Ai-1,j** |  |
| **Ai,j-1** | **Ai,j** | **Ai,j+1** |
|  | **Ai+1,j** |  |

Рахувати кількість і заповнювати допоміжний масив, де порядковий номер відповідає за розмір плями.

**Програма**

|  |
| --- |
| C++ |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("oil.dat");ofstream out("oil.sol");int a[102][102];int b[101];int k,s;void pr(int x,int y){a[x][y]=2; s++;if (a[x-1][y]==1)pr(x-1,y);if (a[x+1][y]==1)pr(x+1,y);if (a[x][y-1]==1)pr(x,y-1);if (a[x][y+1]==1)pr(x,y+1);}void main() {int i,j,n,m;inp>>n>>m;for(i=0;i<=n+1;i++)for(j=0;j<=m+1;j++) a[i][j]=0;for(i=0;i<=n+1;i++)b[i]=0;for(i=1;i<=n;i++)for(j=1;j<=m;j++) inp>>a[i][j];k=0; s=0;for(i=1;i<=n;i++)for(j=1;j<=m;j++)if (a[i][j]==1){b[s]++;k++;s=0;pr(i,j);}out<<k<<endl;for(i=1;i<=n;i++)if (b[i]!=0) out<<i<<" "<<b[i]<<endl;} |

## Задача 5. «Прямокутники»

**Ім’я файлу програми: RECTANGLE.\***

**Ім’я вхідного файлу: RECTANGLE.DAT**

**Ім’я вихідного файлу: RECTANGLE.SOL**

**Максимальний час роботи на одному тесті: 1с**

Шоколадну плитку спочатку розламали N разів, потім кожну утворену частину розділили M разів. Визначити загальну кількість утворених прямокутників.

*Вхідні дані.* Вхідний текстовий файл містить єдиний рядок з двох чисел розділених пропуском (0<=N,M<=2147483647).

*Вихідні дані.* Вихідний текстовий файл містить єдине ціле число – кількість прямокутників.

|  |  |
| --- | --- |
| RECTANGLE.DAT | RECTANGLE.SOL |
| 1 1 | 4 |

Приклад файлів

|  |
| --- |
| **C++** |
| #include "fstream"using namespace std; ifstream inp("rectangle.dat");ofstream out("rectangle.sol");int main(){\_\_int64 r,n,m;inp>>n>>m;r=(n+1)\*(m+1);out<<r<<endl;} |

##  Задача 6. «Квадрат»

**Ім’я файлу програми: square.\***

**Ім’я вхідного файлу: square.DAT**

**Ім’я вихідного файлу: square.SOL**

**Максимальний час роботи на одному тесті: 1с**

Дано цілі числа N та M, які задають розмір шоколадної плитки. Визначити найменшу кількість частинок квадратної форми на яку можна поділити плитку.

*Вхідні дані.* Вхідний текстовий файл містить єдиний рядок з двох чисел розділених пропуском (0<N,M<=264).

*Вихідні дані.* Вихідний текстовий файл містить єдине ціле число – кількість квадратів.

Приклад файлів

|  |  |
| --- | --- |
| SQUARE.DAT | SQUARE.SOL |
| 2 4 | 2 |

**Програма**

|  |
| --- |
| **C++** |
| #include "fstream"using namespace std; ifstream inp("SQUARE.dat");ofstream out("SQUARE.sol");int main(){\_\_int64 rez,n,m;inp>>n>>m;rez=0;while (n>0 && m>0){ if( n>m) {rez=rez+n/m;n=n%m;} else {rez=rez+m/n;m=m%n;}}out<<rez<<endl;} |

## Завдання 7. «Ламана»

**Ім’я файлу програми: LAMAN.\***

**Ім’я вхідного файлу: LAMAN.DAT**

**Ім’я вихідного файлу: LAMAN.SOL**

**Максимальний час роботи на одному тесті: 5с**

Шоколадна плитка являє собою сітку з горизонтальних та вертикальних ліній, точки якої в декартовій системі координат на площині позначено точками з цілими координати. Потрібно поділити шоколадну плитку наступним чином:

* починати з лівого нижнього кута, який знаходиться в початку координат;
* можна пересуватися вздовж цих прямих;
* при проходженні через точку завжди змінювати напрям швидкості на перпендикулярний.

Знайти мінімальну довжину шляху до верхньої правої точки.

Технічні умови. Програма Laman читає з файлу розміри шоколадної плитки (цілі числа, не більші 10^100000). Числа розділено пропуском. Програма виводить на екран єдине число - шукану величину.

**Приклади файлів**

|  |  |
| --- | --- |
| LAMAN.DAT | LAMAN.SOL |
| 2 3 | 5 |

|  |
| --- |
| **C++** |
| #include "fstream"#include "iostream"#include "vector"using namespace std; ifstream inp("laman.dat");ofstream out("laman.sol");int main(){\_\_int64 rez,n,m;int a[10000],b[10000],c[10000],os,max,i,j;char s[10000]; inp>>s;a[0]=strlen(s);for (i=1;i<=a[0];i++) {a[a[0]-(i-1)]=s[i-1]-48;}//for (i=1;i<=a[0];i++)cout<<a[i];cout<<endl;inp>>s;b[0]=strlen(s);for (i=1;i<=b[0];i++) {b[b[0]-(i-1)]=s[i-1]-48;}//for (i=1;i<=b[0];i++)cout<<b[i];cout<<endl;if (a[0]>b[0]) max=1;if (b[0]>a[0]) max=2;if (a[0]==b[0]) {i=a[0];while (a[i]==b[i]) i=i-1;if (a[i]>b[i]) max=1;if (b[i]>a[i]) max=2;}//multif (max==1) { os=0;c[0]=a[0];for (i=1;i<=c[0];i++){c[i]=(a[i]\*2+os)%10;os=(a[i]\*2+os)/10;}if(os>0){c[0]=c[0]+1;c[c[0]]=os;}}//if (max==2){os=0;c[0]=b[0];for(i=1;i<=c[0];i++){c[i]=(b[i]\*2+os)%10;os=(b[i]\*2+os)/10;}if (os>0){c[0]=c[0]+1;c[c[0]]=os;}}if (a[1]%2==0&&b[1]%2==1 || a[1]%2==1 && b[1]%2==0){if (c[1]>0)c[1]=c[1]-1;else {i=1;while (c[i]==0){c[i]=9;i++;}c[i]=c[i]-1;while (c[c[0]]==0)c[0]--;}} for (i=c[0];i>=1;i--)out<<c[i]; out<<endl; } |

## Задача 8. «Білі плями»

**Ім’я файлу програми: WHITE.\***

**Ім’я вхідного файлу: WHITE.DAT**

**Ім’я вихідного файлу: WHITE.SOL**

**Максимальний час роботи на одному тесті: 1с**

Кондитерська фабрика випустила чорно-білий шоколад. Програмістам доручили порахувати вміст білого шоколаду. Співробітники фабрики зберігають інформацію про шоколадну плитку в комп'ютері у вигляді матриці розмірністю *M* x*N*. Комірка матриці містить 0, якщо шоколад чорний та 1, якщо білий (2*≤ M, N ≤ 100)*. У матриці комірки входження білого шоколаду не можуть дотикатися одна до одної ні сторонами, ні кутами.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | **0** | **0** |
|   | 0 | 1 | **1** | **0** |
| 1 | 0 | 0 | **0** | **0** |
| 1 | 0 | 0 | **0** | **1** |
| 1 | 0 | 1 | 0 | 1 |

*Завдання.* Написати програму, яка знаходитиме загальну кількість плям білого шоколаду та кількість плям з однаковою площею.

*Вхідні дані.* Вхідний текстовий файл WHITE.DAT містить в першому рядку два числа *M* та*N,* далі слідують *M* рядків, у кожному по *N* цілих чисел розділених пропусками – елементи матриці.

*Вихідні дані.* Вихідний текстовий файл WHITE.SOL містить у першому рядку ціле число *k -* загальну кількість білих плям, далі у кожному з рядів міститься по два числа, перше – площа плями білого шоколаду, друге – їх кількість. Дані посортувати по площах в порядку зростання.

Приклади файлів

|  |  |
| --- | --- |
| WHITE.DAT | WHITE.SOL |
| 5 51 0 1 0 00 0 1 1 01 0 0 0 01 0 0 0 11 0 1 0 1 | 51 22 13 2 |

**Програма**

|  |
| --- |
| C++ |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("white.dat");ofstream out("white.sol");int a[102][102];int b[101];int k,s;void pr(int x,int y){a[x][y]=2; s++;if (a[x-1][y]==1)pr(x-1,y);if (a[x+1][y]==1)pr(x+1,y);if (a[x][y-1]==1)pr(x,y-1);if (a[x][y+1]==1)pr(x,y+1);}void main() {int i,j,n,m;inp>>n>>m;for(i=0;i<=n+1;i++)for(j=0;j<=m+1;j++) a[i][j]=0;for(i=0;i<=n+1;i++)b[i]=0;for(i=1;i<=n;i++)for(j=1;j<=m;j++) inp>>a[i][j];k=0; s=0;for(i=1;i<=n;i++)for(j=1;j<=m;j++)if (a[i][j]==1){b[s]++;k++;s=0;pr(i,j);}out<<k<<endl;for(i=1;i<=n;i++)if (b[i]!=0) out<<i<<" "<<b[i]<<endl;} |

## Задача 9. Цегляна стіна

**Ім’я вхідного файлу: BRICK.DAT**

**Ім’я вихідного файлу: BRICK.SOL**

**Максимальний час роботи на одному тесті: 1с**

Щоб розділити царство царя Гороха необхідно побудувати цегляну стіну висотою 2 і завдовжки n (висота цеглини дорівнює 2, ширина – 1).

Ваше завдання - написати програму, яка визначає, скільки шаблонів, можливо отримати для стіни завдовжки n.



Вхідний файл містить одне ціле число N (1≤N≤32767).

Ваша програма повинна вивести в вихідний файл рядок, що містить одне ціле число, рівне кількості шаблонів, які можна отримати для стіни завдовжки n.

**Приклад**

|  |  |
| --- | --- |
| BRICK.DAT | BRICK.SOL |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"#include "fstream"using namespace std;void \_tmain(){ifstream inp;inp.open ("brick.dat");ofstream out;out.open("brick.sol");int i,n,j,os;int a[10000],b[10000],c[10000]; inp>>n; for(i=0;i<=9999;i++){a[i]=0;b[i]=0;c[i]=0;}a[0]=1;a[1]=1;b[0]=1;b[1]=2;for (j=3;j<=n;j++){if (a[0]>b[0]) c[0]=a[0]; else c[0]=b[0];os=0;for(i=1;i<=c[0];i++){c[i]=(a[i]+b[i]+os)%10;os=(a[i]+b[i]+os)/10;}if (os>0){c[0]=c[0]+1;c[c[0]]=os;}for (i=0;i<=b[0];i++)a[i]=b[i];for (i=0;i<=c[0];i++)b[i]=c[i];}for (i=c[0];i>=1;i--)out<<c[i];out<<"\n";inp.close();out.close();} |

## Задача 10. Порядок

**Максимальна оцінка: 20 балів**

**Обмеження на час: 5 сек.**

**Обмеження на пам’ять: 32 MБ**

**Вхідний файл: order.dat**

**Вихідний файл: order.sol**

**Програма: order.\***

Нехай *n* — довільне натуральне число, а послідовність *i*1, *i*2, ... , *in* містить усі натуральні числа від 1 до *n* включно. Порушенням порядку у такій послідовності називають систему таких двох нерівностей, що справджуються: *j* < *k* та *ij* > *ik*. Якщо послідовність зростає, то кількість порушень порядку дорівнює 0. Якщо послідовність спадає, то така кількість дорівнює *n*(*n* – 1)/2. В усіх інших випадках ця кількість розташована між вказаними величинами.

 **Завдання**

Встановіть парність кількості порушень порядку послідовності.

**Вхідні дані**

У першому рядку вхідного файлу вказано кількість послідовностей *m*. Кожний з наступних *m* рядків містить натуральне число *n* і послідовність різних натуральних чисел від 1 до *n* включно: *i*1, *i*2, ... , *in* при 2 ≤ *т* ≤ 100, 2 ≤ *n* ≤ 1 048 576.У 50 % тестів *n* ≤ 4096.

**Вихідні дані**

Єдиний рядок вихідного файлу має містити число в шістнадцятковій системі числення, яке відповідає двійковому числу яке утворене з *m* символів — нулів або одиниць — без пропусків: *k*-й символ рядка — це залишок від ділення на 2 кількості порушень порядку *k*-ї послідовності, заданої (*k* + 1)-м рядком вхідного файлу.

**Приклад**

|  |  |
| --- | --- |
| **order.dat** |  **orderd.sol**  |
| 53 1 2 33 2 3 13 1 3 24 2 3 4 14 3 4 1 2 | 6 |

***Пояснення (00110 )2=(6)16***

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"#include "fstream"using namespace std;ifstream inp ("order.dat");ofstream out ("order.sol");int \_tmain(){  int a[104857],b[110];char c[110]; int j,n,k,m,l,x,i; k=0;l=4; for(i=0;i<=110;i++)b[i]=0;inp>>m; for (x=1;x<=m;x++) { inp>>n; for(i=1;i<=n;i++)inp>>a[i]; //for(i=1;i<=n;i++)cout<<a[i]<<" ";cout<<"\n"; k=0; for (j=1;j<=n-1;j++) for(i=j+1;i<=n;i++)if(a[j]>a[i])k++; l++; b[l]=k%2; }//for(i=1;i<=l;i++)cout<<b[i];//перевести двійкове число в 16 i=l;x=0;while (i>4){k=b[i]+b[i-1]\*10+b[i-2]\*100+b[i-3]\*1000;x++;if (k==0) c[x]='0';if (k==1) c[x]='1';if (k==10) c[x]='2';if (k==11) c[x]='3';if (k==100) c[x]='4';if (k==101) c[x]='5';if (k==110) c[x]='6';if (k==111) c[x]='7';if (k==1000) c[x]='8';if (k==1001) c[x]='9';if (k==1010) c[x]='A';if (k==1011) c[x]='B';if (k==1100) c[x]='C';if (k==1101) c[x]='D';if (k==1110) c[x]='E';if (k==1111) c[x]='F';i=i-4;}// відкинути 0 на початкуi=x;while(c[i]=='0')i--;for (j=i;j>=1;j--)out<<c[j]; out<<"\n";} |

## Задача 11. Код Грея

**Максимальна оцінка: 100 балів**

**Обмеження на час: 1 сек.**

**Обмеження на пам’ять: 32 MБ**

**Вхідний файл: grey.dat**

**Вихідний файл: grey.sol**

**Програма: grey.\***

Кодом Грея називають непозиційну систему запису цілих натуральних чисел за допомогою двох символів 0 та 1 таким чином. Нуль кодують послідов­ністю нулів. При зростанні цілого числа:

* наймолодший 1-й розряд у послідовності символів змінюють у такій послі­дов­ності: 0, 1, після чого у наступний 2-й розряд записують 1, а наймолодший розряд змінюють уже у протилежному порядку;
* два наймолодші розряди змінюють у такому порядку: 00, 01, 11, 10, після чого у 3-й розряд записують 1, а два наймолодші розряди змінюють уже у протилежному порядку: 10, 11, 01, 00 ... ;
* *k* наймолодших розрядів змінюють у порядку, визначеному попередніми кро­ка­ми, після чого у наступний (*k* + 1)-й розряд записують 1, а молодші розряди змінюють уже у протилежному порядку.

Коди Грея довжини 4 чисел від 0 до 15 включно такі (записано у порядку зростання числа): 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

Коди Грея двох послідовних натуральних чисел відрізняються лише в одно­му розряді. Це використовують для збільшення надійності роботи системи оптич­них фотодіодів при встановленні кута повороту дисків — носіїв інформації.

**Завдання**

Визначте код Грея натурального числа.

**Вхідні дані**

Вхідний файл містить лише десятковий запис натурального числа *n* при *n <* 1018.

**Вихідні дані**

Вихідний файл має містити код Грея числа *n* мінімальної довжини. Інакше кажучи, цей код має починатися з одиниці й містити *j* символів,

де 2*j* – 1 ≤ *n* < 2*j*.

**Приклади**

|  |  |
| --- | --- |
| **grey.dat** | **grey.sol** |
| 2 | 11 |
| 7 | 100 |
| 13 | 1011 |

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("grey.dat");ofstream out("grey.sol");\_\_int64 g (\_\_int64 n) {return n ^ (n >> 1);}void main(){ int a; inp>>a; \_\_int64 b=g(a); int k,c[100]; k=0;while (b>0){k++;c[k]=b%2;b=b/2;}for (int i=k;i>=1;i--)out<<c[i];out<<"\n";} |

## Задача 12. Паліндроми

Ім’я вхідного файлу: palind.in

Ім’я вихідного файлу: palind.out

Програма: palindr.\*

Обмеження часу: 5с

Обмеження пам’яті: 64 мбайт

Паліндром – слово, яке однаково читається в обох напрямках. Підрахуйте, скільки різних паліндромів можна отримати, переставляючи букви в заданому слові. Так як відповідь може бути дуже великим числом – виведіть остачу від його ділення на 109.

**Формат вхідного файлу**

Один рядок містить слово із рядкових букв латинського алфавіту довжиною від 1 до 100 символів.

**Формат вихідного файлу**

Вихідний файл повинен містити одне ціле число від 0 до 109-1 – відповідь до задачі.

Приклад

|  |  |
| --- | --- |
| palind.in | palind.out |
| ababc | 2 |
| aaa | 1 |
| abc | 0 |

В першому прикладі можна зробити два паліндроми: abcba, bacab

**Програма**

|  |
| --- |
| **С++** |
| #include <fstream>#include "iostream"using namespace std;ifstream inp ("palind.in");ofstream out ("palind.out");\_\_int64 fac(int n){\_\_int64 t=1;for (int i=1;i<=n;i++)t=(t\*i);return t;}int \_tmain(){int k,b[36];for (int i=1; i<=35;i++) b[i]=0;  char a[100]; inp>>a; for (int i=0; i<strlen(a);i++) {k=a[i];b[k-96]=b[k-96]+1; }int p=0;for (int i=1; i<=35;i++) if (b[i]%2==1)p++;if (p>1){out<<0<<"\n";exit(0);}int rez=1; for (int i=1; i<=35;i++) {rez=rez\*fac(b[i]/2);} int rez1=fac(strlen(a)/2)/rez;out<<rez1<<"\n";} |

## Задача 13. Рядки

Ім’я вхідного файлу: string.in

Ім’я вихідного файлу: string.out

Програма: strings.\*

Обмеження часу: 4с

Обмеження пам’яті: 64 мбайт

Маємо два рядка. Із кожного рядка дозволяється видаляти символи, але кількість видалених символів, які йдуть підряд, не повинна перевищувати W. Ваше завдання – видаливши мінімально можливу кількість символів, зробити рядки однаковими (символи різного регістру вважати однаковими).

**Формат вхідного файлу**

Вхідний файл містить в першому рядку число W (1≤W≤1500), в другому і третьому – два заданих рядка, які складаються із цифр і символів англійського алфавіту від 1 до 1500 символів.

**Формат вихідного файлу**

Вихідний файл повинен містити один рядок, який можна отримати із обох рядків за правилами задачі. Якщо існує декілька варіантів відповіді, виведіть будь-який. Якщо відповіді не існує виведіть No solution

* **Приклад**

|  |  |
| --- | --- |
| * String.in
 | * String.out
 |
| * 1xabcdaefdz
 | * No solution
 |
| * 2xabcdaefdz
 | * ad
 |

**Програма**

|  |
| --- |
| **С++** |
| #include <iostream>#include <vector>#include <string>#include <fstream>using namespace std;ifstream inp("string.dat");ofstream out("syring.sol"); string getLongestCommonSubsequence(const string& a, const string& b) { vector<vector<int> > max\_len; max\_len.resize(a.size() + 1); for(int i = 0; i <= static\_cast<int>(a.size()); i++) max\_len[i].resize(b.size() + 1); for(int i = static\_cast<int>(a.size()) - 1; i >= 0; i--) { for(int j = static\_cast<int>(b.size()) - 1; j >= 0; j--) { if(a[i] == b[j]) { max\_len[i][j] = 1 + max\_len[i+1][j+1]; } else { max\_len[i][j] = max(max\_len[i+1][j], max\_len[i][j+1]); } } } string res; for(int i = 0, j = 0; max\_len[i][j] != 0 && i < static\_cast<int>(a.size()) && j < static\_cast<int>(b.size()); ) { if(a[i] == b[j]) { res.push\_back(a[i]); i++; j++; } else { if(max\_len[i][j] == max\_len[i+1][j]) i++; else j++; } } return res; }int \_tmain(){ string s1; string s2; inp >>s1>>s2; int w=10;string s3=getLongestCommonSubsequence(s1,s2); string s4=s3; int n=s3.length(); if (n==0) s4="No solution"; else {int j=0;int k=0;for (int i=0; i<n;i++){k=0;while (s3[i]!=s1[j]&&j<s1.length()) {j++;k++;}j++;if (k>w) s4="No solution";}k=s1.length()-j;if (k>w) s4="No solution";n=s3.length();//s2j=0;k=0;for (int i=0; i<n;i++){k=0;while (s3[i]!=s2[j]&& j<s2.length()) {j++;k++;}j++;if (k>w) s4="No solution";}k=s2.length()-j;if (k>w) s4="No solution"; }out<<s4<<"\n";} |

## ****Задача 14. Сума****

Ім’я вхідного файлу: suma.in

Ім’я вихідного файлу: suma.out

Програма: suma.\*

Обмеження часу: 1 с

Обмеження пам’яті: 16 мбайт

Знайти суму двох цілих чисел.

**Формат вхідного файлу**

У двох рядках записані цілі числа, модуль кожного з яких не більше за 10^1000 . Перед від'ємними числами стоїть знак "мінус", пред додатними - нічого не стоїть.

**Формат вихідного файлу**

Вихідний файл повинен містити одне ціле число – відповідь до задачі.

Приклад

|  |  |
| --- | --- |
| suma.in | suma.out |
| 12 | 3 |
| -53 | -2 |
| -4-3 | -7 |

**Програма**

|  |
| --- |
| **С++** |
| **#include "iostream"****#include "fstream"****using namespace std;****ifstream inp("suma.in");****ofstream out("suma.out");****int main()****{** **char z1,z2,z3;****int n1,n2,n3,i,m,os;** **char s1[1001],s2[1001];** **int a[1001], b[1001],c[1001];** **for (i=0;i<1001;i++)a[i]=0;** **for (i=0;i<1001;i++)b[i]=0;** **for (i=0;i<1001;i++)c[i]=0;** **inp>>s1;** **n1=strlen(s1);** **if(s1[0]=='-'){** **z1='-';****for (int i=1;i<n1;i++)****a[n1-i-1]=s1[i]-48;** **n1=n1-1;** **}** **else {z1='+';** **for (int i=0;i<n1;i++)** **a[n1-i-1]=s1[i]-48;}** **inp>>s2;** **n2=strlen(s2);** **if(s2[0]=='-'){z2='-';****for (int i=1;i<n2;i++)****b[n2-i-1]=s2[i]-48;** **n2=n2-1;** **}** **else {z2='+';** **for (int i=0;i<n2;i++)** **b[n2-i-1]=s2[i]-48;}****//cout<<z1; for (int i=0;i<n1;i++)cout<<a[i]; cout<<"\n";****//cout<<z2; for (int i=0;i<n2;i++)cout<<b[i];cout<<"\n";****if (z1=='+'&& z2=='+' ||z1=='-'&& z2=='-' )** **{****if (z1=='-'&& z2=='-') z3='-';else z3='+';****if (n1>n2) n3=n1; else n3=n2;****os=0;****for(i=0;i<n3;i++)****{****c[i]=(a[i]+b[i]+os)%10;****os=(a[i]+b[i]+os)/10;****}****if (os>0){n3=n3+1;c[n3-1]=os;}****}****if (z1=='+'&& z2=='-' ||z1=='-'&& z2=='+' )****{** **if (n1>n2) m=1;****if (n2>n1) m=2;****if (n2==n1)****{****i=n2-1; while(a[i]==b[i] && i>0)i--;****if (a[i]>b[i]) m=1;****if (a[i]<b[i]) m=2;****if (a[i]==b[i]) m=3;****}****if (m==1)****{****z3=z1;****n3=n1;****for(i=0;i<n1;i++)****if (a[i]>=b[i]) c[i]=a[i]-b[i];****else {c[i]=10+a[i]-b[i];a[i+1]=a[i+1]-1;}****while (c[n3-1]==0&&n3>1)n3=n3-1;****}****if (m==2)****{** **z3=z2;****n3=n2;****for(i=0;i<n2;i++)****if (b[i]>=a[i]) c[i]=b[i]-a[i];****else {c[i]=10+b[i]-a[i];b[i+1]=b[i+1]-1;}****while (c[n3-1]==0&&n3>1)n3=n3-1;** **}****if (m==3)****{** **z3='+';** **c[0]=0;** **n3=1;****}****}****if (z3=='-')out<<z3;** **for (int i=n3-1;i>=0;i--)out<<c[i];out<<"\n";****}** |

## ****Задача 15. Квадратний корінь****

Ім’я вхідного файлу: korin.in

Ім’я вихідного файлу: korin.out

Програма: korin.\*

Обмеження часу: 2 с

Обмеження пам’яті: 64 мбайт

Для заданого натурального числа **А** потрібно знайти найбільше число **В** таке, що **B^2** ≤ **A**.

**Технічні умови**

**Вхідні дані**

У вхідному файлі записано натуральне число **A** (**A** ≤ **10^3000**).

**Вихідні дані**

У вихідний файл виведіть максимальне натуральне число **B**, квадрат якого не перевищує **A**. Число **B** слід виводити без лідируючих нулів.

Приклад

|  |  |
| --- | --- |
| Приклад вхідних даних27 | Приклад вихідних даних5 |

 **Програма**

|  |
| --- |
| **С++** |
| #include "iostream"#include "fstream"using namespace std;ifstream inp("suma.in");ofstream out("suma.out");int main(){ int max,na,nb,nc,nx,nd,np,i,os,j,temp,nd1; int a[3002], b[3002],c[3002],x[3002],d[3002],p[3002],d1[3002];  for (i=0;i<3001;i++)a[i]=0; for (i=0;i<3001;i++)b[i]=0; for (i=0;i<3001;i++)c[i]=0; for (i=0;i<3001;i++)x[i]=0; for (i=0;i<3001;i++)d[i]=0; for (i=0;i<3001;i++)d1[i]=0;  char s[3002]; inp>>s;  na=1;a[0]=1; nb=strlen(s);for(i=0;i<nb;i++)b[nb-i-1]=s[i]-48;np=nb;for(i=0;i<nb;i++)p[i]=b[i];nc=0;nd=0;while(c[0]!=1 || nc!=1){for (i=3001;i>=0;i--)d[i]=0;nd=0;for (i=3001;i>=0;i--)c[i]=0;nc=0;// summanc=nb;os=0;for(i=0;i<nc;i++){c[i]=(a[i]+b[i]+os)%10;os=(a[i]+b[i]+os)/10;}if (os>0){nc=nc+1;c[nc-1]=os;}//---------------// ділення пополам os=0;nx=nc; for(i=nx-1;i>=0;i--){x[i]=(c[i]+os\*10)/2;os=(c[i]+os\*10)%2;} if (x[nx-1]==0)nx--;// множенняnd=0;nc=nx; for(j=0;j<nx;j++) { os=0;for(i=0;i<nc;i++){c[i]=(x[i]\*x[j]+os)%10;os=(x[i]\*x[j]+os)/10;}if (os>0){nc=nc+1;c[nc-1]=os;}else nc++;for(i=nc-1;i>=0;i--)c[i+j]=c[i];for(i=0;i<j;i++)c[i]=0;//cout<<j<<"-------";for (i=nc-1;i>=0;i--)cout<<c[i];cout<<"\n";if (nc>=nd)nd=nc;os=0;for(i=0;i<nd;i++){temp=(c[i]+d[i]+os)%10;os=(c[i]+d[i]+os)/10;d[i]=temp;}if (os>0){nd=nd+1;d[nd]=os;} }if (d[nd-1]==0)nd=nd-1;//cout<<"kvad "; for (i=nd-1;i>=0;i--)cout<<d[i];cout<<"\n";//cout<<"xxx ";for (i=nx-1;i>=0;i--)cout<<x[i];cout<<"\n";// порівнянняif (nd>np) max=0;if (np>nd) max=1;if (np==nd){i=nd-1; while(p[i]==d[i]&&i>0)i--;if (d[i]>p[i]) max=0; if (d[i]<p[i]) max=1; if (d[i]==p[i]) max=2; }if (max==0||max==2){nb=nx;for(i=0;i<nb;i++)b[i]=x[i];}if (max==1){na=nx;for(i=0;i<na;i++)a[i]=x[i];}//vidnimannand1=nb;for (i=0;i<=nd1;i++)d1[i]=b[i];nc=nb;for (i=0;i<=nc;i++)c[i]=0;for(i=0;i<nd1;i++)if (d1[i]>=a[i]) c[i]=d1[i]-a[i];else {c[i]=10+d1[i]-a[i];d1[i+1]=d1[i+1]-1;}while (c[nc-1]==0)nc=nc-1;//------------//cout<<"max="<<max<<"\n"; //for (i=nc-1;i>=0;i--)cout<<c[i];cout<<"\n"; //if (max==0)for (i=na-1;i>=0;i--)cout<<a[i];//if (max==1)for (i=nb-1;i>=0;i--)cout<<b[i];cout<<"\n"; //cin>>s;}int max1,max2;for (i=3001;i>=0;i--)d[i]=0;nd=0;for (i=3001;i>=0;i--)c[i]=0;nc=0;// множенняnx=na;for (i=0;i<na;i++)x[i]=a[i];nd=0;nc=nx; for(j=0;j<nx;j++) { os=0;for(i=0;i<nc;i++){c[i]=(x[i]\*x[j]+os)%10;os=(x[i]\*x[j]+os)/10;}if (os>0){nc=nc+1;c[nc-1]=os;}else nc++;for(i=nc-1;i>=0;i--)c[i+j]=c[i];for(i=0;i<j;i++)c[i]=0;//cout<<j<<"-------";for (i=nc-1;i>=0;i--)cout<<c[i];cout<<"\n";if (nc>=nd)nd=nc;os=0;for(i=0;i<nd;i++){temp=(c[i]+d[i]+os)%10;os=(c[i]+d[i]+os)/10;d[i]=temp;}if (os>0){nd=nd+1;d[nd]=os;} }if (d[nd-1]==0)nd=nd-1;//cout<<"kvad "; for (i=nd-1;i>=0;i--)cout<<d[i];cout<<"\n";//cout<<"xxx ";for (i=nx-1;i>=0;i--)cout<<x[i];cout<<"\n";// порівнянняif (nd>np) max1=0;if (nd<np) max1=1;if (np==nd){i=nd-1; while(p[i]==d[i]&&i>0)i--;if (d[i]>p[i]) max1=0; if (d[i]<p[i]) max1=1; if (d[i]==p[i]) max1=2; }for (i=3001;i>=0;i--)d[i]=0;nd=0;for (i=3001;i>=0;i--)c[i]=0;nc=0;// множенняnx=nb;for (i=0;i<nb;i++)x[i]=b[i];nd=0;nc=nx; for(j=0;j<nx;j++) { os=0;for(i=0;i<nc;i++){c[i]=(x[i]\*x[j]+os)%10;os=(x[i]\*x[j]+os)/10;}if (os>0){nc=nc+1;c[nc-1]=os;}else nc++;for(i=nc-1;i>=0;i--)c[i+j]=c[i];for(i=0;i<j;i++)c[i]=0;//cout<<j<<"-------";for (i=nc-1;i>=0;i--)cout<<c[i];cout<<"\n";if (nc>=nd)nd=nc;os=0;for(i=0;i<nd;i++){temp=(c[i]+d[i]+os)%10;os=(c[i]+d[i]+os)/10;d[i]=temp;}if (os>0){nd=nd+1;d[nd]=os;} }if (d[nd-1]==0)nd=nd-1;//cout<<"kvad "; for (i=nd-1;i>=0;i--)cout<<d[i];cout<<"\n";//cout<<"xxx ";for (i=nx-1;i>=0;i--)cout<<x[i];cout<<"\n";// порівнянняif (nd>np) max2=0;if (nd<np) max2=1;if (np==nd){i=nd-1; while(p[i]==d[i]&&i>0)i--;if (d[i]>p[i]) max2=0; if (d[i]<p[i]) max2=1; if (d[i]==p[i]) max2=2; }//if (max==2){for (i=nx-1;i>=0;i--)out<<x[i];out<<"\n";exit(0);}if (max2==2 || max2==1){for (i=nb-1;i>=0;i--)out<<b[i];out<<"\n";exit(0);} if (max1==2 || max1==1){for (i=na-1;i>=0;i--)out<<a[i];out<<"\n";exit(0);} } |

## Задача 16. MATCHES

**Ім’я вхідного файлу: MATCHES.DAT**

**Ім’я вихідного файлу: MATCHES.SOL**

**Максимальний час роботи на одному тесті: 2с**

Відомо, що за перемогу у матчах чемпіонату з футболу команді нараховується три очки, за нічию – одне очко, за поразку очки не нараховуються.

Необхідно написати програму для знаходження числа всіх можливих варіантів здобуття за N матчів деякою футбольною командою M очок.

*Формат вхідних даних.*

Єдиний рядок вхідного файлу **MATCHES.DAT** містить два натуральні числа N та M (1<=N<=20,0<=M<=60). Числа між собою розділені пробілами.

*Формат вихідних даних.*

Єдиний рядок вихідного файлу **MATCHES.SOL** повинен містити одне натуральне число – кількість всіх можливих варіантів.

Приклад.

**MATCHES.DAT:**

3 3

**MATCHES.SOL:**

4

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"#include "fstream"#include "math.h"using namespace std;ifstream inp("matches.dat");ofstream out("matches.sol");\_\_int64 fac(int n){  \_\_int64 t=1;for (int i=1;i<=n;i++)t=t\*i;return t;}int main(){\_\_int64 rez,n,k,k0,k1,k3;inp>>n>>k;k3=k/3;k1=k%3;k0=n-k1-k3;rez=0;while(k1+k3<=n){rez=rez+fac(n)/(fac(k3)\*fac(k1)\*fac(k0));if (k3>0)k3=k3-1;k1=k1+3;k0=n-k3-k1;}out<<rez<<"\n";} |

## Задача17. POINT

**Ім’я вхідного файлу: POINT.DAT**

**Ім’я вихідного файлу: POINT.SOL**

**Максимальний час роботи на одному тесті: 1с**

Багатокутник (не обов'язково опуклий) на площині заданий координатами своїх вершин. Потрібно підрахувати кількість точок з цілочисельними координатами, що лежать всередині нього (але не на його межі).

**Формат вхідних даних**

У першому рядку міститься N (3 <= N <= 1000) - число вершин багатокутника. У наступних N рядках йдуть координати (Xi, Yi) вершин багатокутника в порядку обходу за годинниковою стрілкою. Xi і Yi - цілі числа, по модулю не перевищують 1000000.

**Формат вихідних даних**

У вихідний файл вивести одне число - шукану кількість точок.

Приклади:

|  |  |
| --- | --- |
| POINT.DAT | POINT.SOL |
| 4 -1 -1-1 11 11 -1 | 1 |
| 30 00 22 0 | 0 |

**Програма**

|  |
| --- |
| **С++** |
| #include <fstream>using namespace std;int MAX(int a1,int a2){if(a1>a2)return a1; return a2;}int MIN(int a1,int a2){if(a1>a2)return a2; return a1;}int NOD(int a1,int a2){int z;if(a1<a2){z=a1;a1=a2;a2=z;}if(a2!=0) while(a2){ z=a2; a2=a1%a2; a1=z; }if(a1<0)a1\*=-1;return a1;}int main(){int N;long long S=0;int sum\_st=0;int x1,y1,x2,y2,px,py;ifstream file\_if("POINT.DAT");ofstream file\_of("POINT.SOL");file\_if>>N;file\_if>>x1;file\_if>>y1;px=x1;py=y1;for(int i=1;i<N;i++){file\_if>>x2; file\_if>>y2; if(x1!=x2 && y1!=y2)sum\_st+=NOD(MAX(x1,x2)-MIN(x1,x2),MAX(y1,y2)-MIN(y1,y2));else if(x1==x2)sum\_st+=(MAX(y1,y2)-MIN(y1,y2));else sum\_st+=(MAX(x1,x2)-MIN(x1,x2));S+=(y2+y1)\*(x2-x1);x1=x2; y1=y2;}x2=px; y2=py; if(x1!=x2 && y1!=y2)sum\_st+=NOD(MAX(x1,x2)-MIN(x1,x2),MAX(y1,y2)-MIN(y1,y2));else if(x1==x2)sum\_st+=(MAX(y1,y2)-MIN(y1,y2));else sum\_st+=(MAX(x1,x2)-MIN(x1,x2));S+=(y2+y1)\*(x2-x1);if(S<0)S\*=-1;file\_of<<(1+(S-sum\_st)/2);file\_if.close();file\_of.close(); return 0;} |

## Задача 18. POLYGON

**Ім’я вхідного файлу: POLYGON.DAT**

**Ім’я вихідного файлу: POLYGON.SOL**

**Максимальний час роботи на одному тесті: 2 с**

На площині задано дві фігури, що обмежені опуклими багатокутниками. Фігури розташовані таким чином, що їх вершини не співпадають, а контури мають рівно 2 точки перетину.

Довільним чином розділимо площину прямою. Будемо вважати, що півплощина з одного боку прямої відповідає першій фігурі, а з іншого боку – другій фігурі. Визначимо поняття дефекту розділення – сума площі першої фігури, що розташована на півплощині другої фігури, та площі другої фігури, що розташована на півплощині першої фігури. З двох можливих відповідностей півплощин до фігур оберемо таку відповідність, де значення дефекту менше.

Наприклад, на рисунку зображена пряма, що задає певне розділення багатокутників. Оцінка дефекту цього розділення (два випадки відповідності): (D)+(C+E) та (A+D)+(B+C). З рисунку, D+C+E менше, отже, загалом, оцінка розбиття дефекту розділення утвореного цією прямою є D+C+E.

Завдання

Напишіть програму polygon, що за заданими двома багатокутниками знаходить пряму, що утворює розділення з найменшим дефектом.

Вхідні дані

Перший рядок вхідного файлу polygon.DAT містить одне ціле число N (3<=N<=20) – кількість вершин першого багатокутника. Наступні N рядків містять пари цілих чисел – координати вершин першого багатокутника у порядку обходу. (N+2)-й рядок файлу містить число M (3<=M<=20) – кількість вершин другого багатокутника. Наступні M рядків містять його координати задані таким же чином, як і для першого багатокутника. Координати точок додатні та не перевищують 1000.

Вихідні дані

Єдиний рядок вихідного файлу polygon.SOL має містити дві пари чисел – координат двох точок, що однозначно задають розділення (пряму) з найменшим дефектом. Числа потрібно виводити у порядку: x1 y1 x2 y2. Координати потрібно виводити з точністю 10-3. Координати точок мають бути додатні та не більші за 1000.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| polygon.DAT | polygon.SOL |
| 32 13 34 135 23 24 3 | 2 5 4 1 |

**Програма**

|  |
| --- |
| **С++** |
| // polygon.cpp: определяет точку входа для консольного приложения.//#include <fstream>using namespace std;int main(){ifstream file\_if("POLYGON.DAT");ofstream file\_of("POLYGON.SOL");int N,M,\*p1[2],\*p2[2];//----------------------Зчитування-------------------------------------------file\_if>>N;p1[0]=new int[N+1]; p1[1]=new int[N+1];for(int i=0;i<N;i++){ file\_if>>p1[0][i]; file\_if>>p1[1][i]; }file\_if>>M;p2[0]=new int[M+1]; p2[1]=new int[M+1];for(int i=0;i<M;i++){ file\_if>>p2[0][i]; file\_if>>p2[1][i]; }//----------------------Замикання-------------------------------------------p1[0][N]=p1[0][0]; p1[1][N]=p1[1][0];p2[0][M]=p2[0][0]; p2[1][M]=p2[1][0];//----------------------Знаходимо точки перетину ребер----------------------double t1,t2;bool ok1,ok2;double x1,y1,x2,y2;int a1,a2,b1,b2,c1,c2;bool xy1=false;for(int i=0;i<N;i++)for(int j=0;j<M;j++){ok1=false; ok2=false;a1=(p1[1][i+1]-p1[1][i]);b1=(p1[0][i]-p1[0][i+1]);c1=(p1[0][i+1]\*p1[1][i]-p1[0][i]\*p1[1][i+1]);t1=a1\*p2[0][j]+b1\*p2[1][j]+c1;t2=a1\*p2[0][j+1]+b1\*p2[1][j+1]+c1;if((t1>0 && t2<0)||(t1<0 && t2>0))ok1=true;a2=(p2[1][j+1]-p2[1][j]);b2=(p2[0][j]-p2[0][j+1]);c2=(p2[0][j+1]\*p2[1][j]-p2[0][j]\*p2[1][j+1]);t1=a2\*p1[0][i]+b2\*p1[1][i]+c2;t2=a2\*p1[0][i+1]+b2\*p1[1][i+1]+c2;if((t1>0 && t2<0)||(t1<0 && t2>0))ok2=true;if(ok1 && ok2){c1\*=-1;c2\*=-1;if(!xy1){ x1=(double)(b2\*c1-b1\*c2)/(a1\*b2-a2\*b1); y1=(double)(a1\*c2-a2\*c1)/(a1\*b2-a2\*b1); xy1=true;}else{ x2=(double)(b2\*c1-b1\*c2)/(a1\*b2-a2\*b1); y2=(double)(a1\*c2-a2\*c1)/(a1\*b2-a2\*b1); break;}}}double zm;if(x1>x2){zm=x1; x1=x2; x2=zm; zm=y1; y1=y2; y2=zm;}else if(x1==x2 && y1>y2){zm=y1; y1=y2; y2=zm;}int k;k=4; if(x1>10)k+=1; if(x1>100)k+=1; file\_of.precision(k);file\_of<<x1;file\_of<<" ";k=4; if(y1>10)k+=1; if(y1>100)k+=1; file\_of.precision(k);file\_of<<y1;file\_of<<" ";k=4; if(x1>10)k+=1; if(x1>100)k+=1; file\_of.precision(k);file\_of<<x2;file\_of<<" ";k=4; if(y2>10)k+=1; if(y2>100)k+=1; file\_of.precision(k);file\_of<<y2;file\_if.close();file\_of.close(); return 0;}//--------------------------------------------------------------------------- |

## Задача 19. Точка

|  |  |
| --- | --- |
| **Обмеження часу:** | 1 с |
| **Обмеження пам’яті:** | 64 M |

Задано точку з координатами х та у. Визначити, в якій координатній чверті вона розміщена.

Формат вхідних даних

З клавіатури вводяться цілі числа х та у, які розділені одним пробілом (-1000 < х, у < 1000).

Формат вихідних даних

На екран виводиться одне число – координатна чверть або 0, якщо визначити однозначно чверть не можна

**Приклади**

|  |  |
| --- | --- |
| **Вхідні дані** | **Результат роботи** |
| -1 -1 | 3 |

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"using namespace std;int main(){int x,y;cin>>x>>y;if (x>0 &&y>0) cout<<1<<endl;elseif (x<0 &&y>0) cout<<2<<endl;elseif (x<0 &&y<0) cout<<3<<endl;elseif (x>0 &&y<0) cout<<4<<endl;elsecout<<0<<endl; return 0;} |

## Задача 20. Стіл

|  |  |
| --- | --- |
| **Обмеження часу:** | 1 с |
| **Обмеження пам’яті:** | 64 M |

На столі лежать дві коробки розмірами A1×B1×C1 та A2×B2×C2. З’ясувати, чи можна одну із цих коробок поставити в іншу, якщо дозволено повороти коробок через будь-яке ребро на кут 90 градусів.

Вхідні дані

З клавіатури вводяться цілі числа A1, B1, C1 та A2, B2, C2. Всі числа натуральні і не більші за 1000.

Вихідні дані

На екран виводиться одне число: 0 – якщо коробки рівні, 1 – якщо другу потрібно помістити в першу, 2 – якщо першу потрібно помістити в другу, -1 – інакше

**Приклади**

|  |  |
| --- | --- |
| **Вхідні дані** | **Результат роботи** |
| 1 2 33 2 1 | 0 |
| 2 2 33 2 1 | 1 |
| 2 2 33 2 3 | 2 |
| 3 4 52 4 6 | -1 |

**Програма**

|  |
| --- |
| **С++** |
| #include <cstdlib>#include "iostream"using namespace std;int main(){int a1,a2,b1,b2,c1,c2,t;cin>>a1>>b1>>c1;cin>>a2>>b2>>c2;if (a1>b1) {t=a1;a1=b1;b1=t;}if (a1>c1) {t=a1;a1=c1;c1=t;}if (b1>c1) {t=b1;b1=c1;c1=t;}if (a2>b2) {t=a2;a2=b2;b2=t;}if (a2>c2) {t=a2;a2=c2;c2=t;}if (b2>c2) {t=b2;b2=c2;c2=t;}if (a1==a2&&b1==b2&&c1==c2) cout<<0<<endl;elseif (a1>=a2&&b1>=b2&&c1>=c2) cout<<1<<endl;elseif (a1<=a2&&b1<=b2&&c1<=c2) cout<<2<<endl;elsecout<<-1<<endl;return 0;} |

##

## Задача 21. Максимум

|  |  |
| --- | --- |
| **Обмеження часу:** | 1 с |
| **Обмеження пам’яті:** | 64 M |

На прямій задано n точок. Координати цих точок x1, x2, ..., xn. Потрібно знайти такі три різні точки i, j та k (xi > xj > xk), щоб значення виразу (xi – xj)\*(xj – xk) було максимальним.

Вхідні дані

Ввести з клавіатури число n (1< n < 105). Далі йдуть n цілих чисел, які по модулю не перевищують 106. У 40% тестів число n не перевищує 100.

Вихідні дані

На екран вивести одне число – максимальне значення виразу.

**Приклади**

|  |  |
| --- | --- |
| **Вхідні дані** | **Результат роботи** |
| 5 3 1 5 2 4 | 4 |

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"using namespace std;int main(){long long n,i,max,min;long long r,rr, a[100002];cin>>n;for(i=1;i<=n;i++)cin>>a[i];max=a[1];min=a[1];for(i=1;i<=n;i++){if (a[i]>max){max=a[i];}if (a[i]<min){min=a[i];}}rr=0;for(i=1;i<=n;i++){ r=(max-a[i])\*(a[i]-min);if (r>rr)rr=r;}cout<<rr<<endl; return 0;} |

## Задача 22. Нумеролог

|  |  |
| --- | --- |
| **Обмеження часу:** | 5 с |
| **Обмеження пам’яті:** | 64 M |

Числом нумеролога для числа n називають таке перетворення. Розкладаємо число n на цифри і знаходимо їх суму. Якщо результат складається більше ніж з однієї цифри, то цю операцію повторюємо до тих пір, поки результатом не стане одна цифра. Отриманий результат і буде числом нумеролога для числа n. Наприклад, числом нумеролога для числа 99 буде 9 (9+9=18, 1+8=9). Визначити, для котрого із заданих двох чисел n та m число нумеролога більше.

Формат вхідних даних

З клавіатури вводяться натуральні числа n та m (1 ≤ n , m ≤ 10300). Кожне число задане в новому рядку.

Формат вихідних даних

На екран виводиться одне число: 0 – якщо числа нумеролога їх рівні, 1 – якщо число нумеролога для першого числа більше за число нумеролога для другого числа, 2 – якщо число нумеролога для другого числа більше за число нумеролога для першого числа.

**Приклади**

|  |  |
| --- | --- |
| **Вхідні дані** | **Результат роботи** |
| 111111134 | 0 |
| 234523456 | 1 |
| 11111111111111 | 2 |

**Програма**

|  |
| --- |
| **С++** |
| #include <cstdlib>#include "iostream"#include "string.h"using namespace std;int main(){ char n[3001],m[3001]; int s1,s2,i,temp; cin>>n; cin>>m; s1=0; int n1=strlen(n); for (i=0;i<n1;i++)s1=s1+(n[i]-48); while (s1>9) {temp=s1; s1=0; while(temp>0) {s1=s1+temp%10; temp=temp/10; } }s2=0;int n2=strlen(m);for (i=0;i<n2;i++)s2=s2+(m[i]-48); while (s2>9) {temp=s2; s2=0; while(temp>0) {s2=s2+temp%10; temp=temp/10; } }//cout<<strlen(n)<<" "<<s2<<endl; if(s1>s2) cout<<"1"<<"\n"; else if(s1<s2)cout<<"2"<<"\n"; else cout<<"0"<<"\n"; return 0;} |

## Задача 23. Спіраль

|  |  |
| --- | --- |
| **Обмеження часу:** | 2 с |
| **Обмеження пам’яті:** | 64 M |

Спіраль – це ламана не нульової довжини без самоперетинів, вершини якої розташовані у точках з цілими координатами на площині. Кожна наступна ланка ламаної повинна бути повернута відносно попередньої на 90º за годинниковою стрілкою. Розглянемо прямокутник n на m. Нехай для кожної спіралі її перша вершина співпадає з лівою верхньою вершиною прямокутника, а друга лежить на верхній стороні прямокутника. Напишіть програму, яка визначає кількість таких спіралей, що лежать в межах даного прямокутника.

Вхідні дані

З клавіатури вводяться натуральні числа n та m (1 ≤ n , m ≤ 20). Гарантовано є тести 2х3, 3х3 та 3х4.

Вихідні дані

На екран вивести єдине число – кількість можливих спіралей.



**Приклади**

|  |  |
| --- | --- |
| **Вхідні дані** | **Результат роботи** |
| 2 2 | 16 |

**Програма**

|  |
| --- |
| **С++** |
| #include "iostream"using namespace std;int main(){long long int a[21][21]; int i,j,x,y;cin>>x>>y; a[1][1]=3; for (i=2;i<=20;i++){ a[1][i]=a[1][i-1]+i+2; a[i][1]=a[i-1][1]+i+1; } for (i=2;i<=20;i++)for (j=2;j<=20;j++) a[i][j]=a[i][j-1]+a[i-1][j]+i+1; cout<<a[x][y]<<endl; return 0;} |